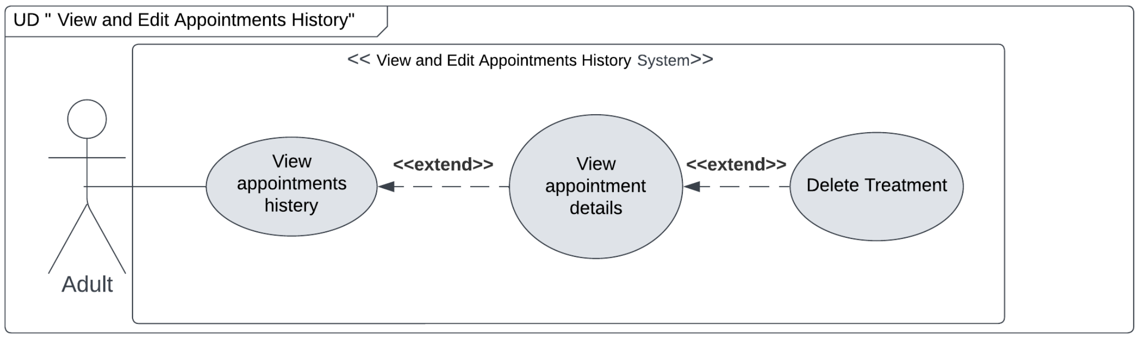
Hospital System

1. User requirements:

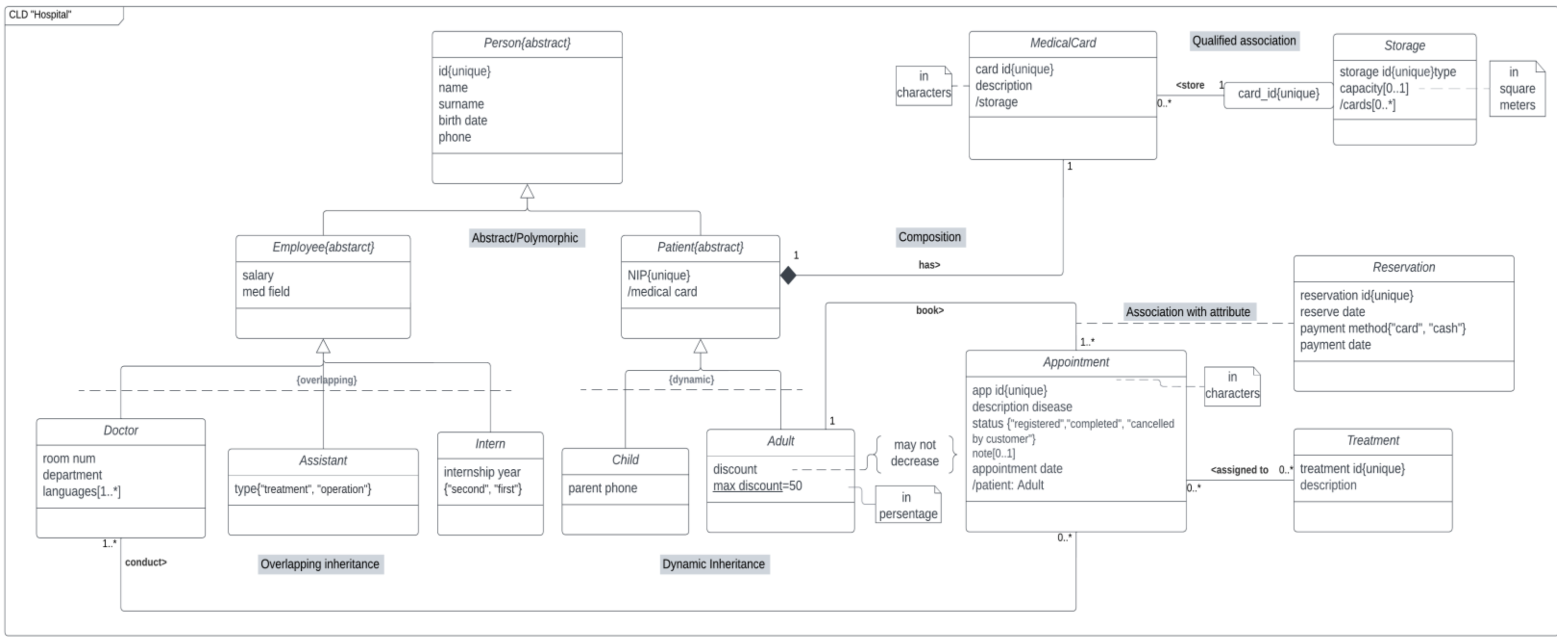
- The system should allow the patient, who is an Adult, view and edit Appointment History. For that the system displays the “Appointments History” page with the list of all previous appointments made by this Adult patient.
- The Adult should be able to choose appointment from previous appointments list on the “Appointments History” page.
- The system should be configured to redirect Adult patient from “Appointment History” page to “Appointment Details” page.
- The Adult should be allowed to view details of chosen appointment on the "Appointment Details" page. To see appointment info like: id, status, description, note, date; appointment reservation info: reservation id and date, payment date and method; treatments assigned to this appointment: id and description.
- The Adult should be able to press on buttons like one "Back" and "Delete" on the "Appointment Details" page.
- The system should be configured to redirect Adult patient from “Appointment Details” page to “Appointment History” page if Adult would press “Back” button.
- The system should be configured to delete chosen treatment from database if Adult would press “Delete” button; and refresh the view of the "Appointment Details" page.

2. Use Case Diagram:

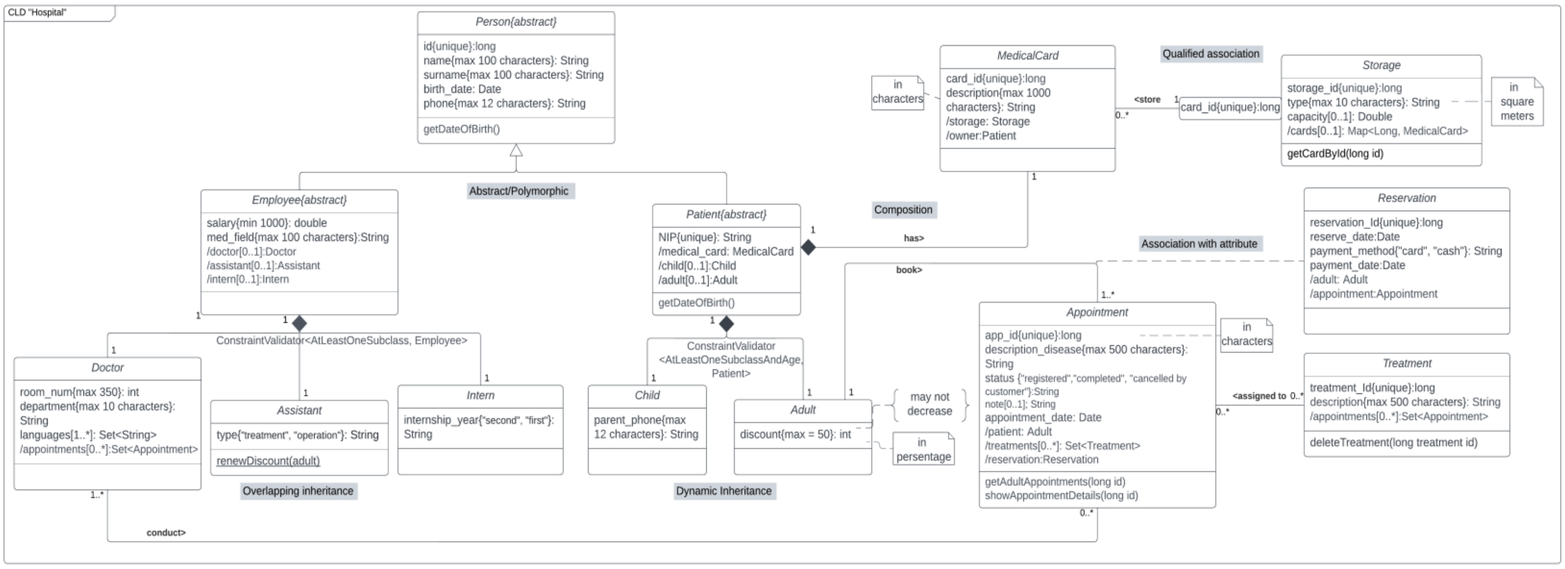

UD " View and Edit Appointments History"
<< View and Edit Appointments History System>>
Adult
View appointments histery
<<extend>>
View appointment details
<<extend>>
Delete Treatment

---

3. Class Diagram – analytical:



4. Class Diagram – design:


CLD "Hospital"
Person{abstract}
id{unique}:long
name{max 100 characters}: String
surname{max 100 characters}: String
birth_date: Date
phone{max 12 characters}: String
getDateOfBirth()
Abstract/Polymorphic
Employee{abstract}
salary{min 1000}: double
med_field{max 100 characters}:String
/doctor[0..1]:Doctor
/assistant[0..1]:Assistant
/intern[0..1]:Intern
ConstraintValidator<AtLeastOneSubclass, Employee>
Doctor
room_num{max 350}: int
department{max 10 characters}: String
languages[1..*]: Set<String>
/appointments[0..*]:Set<Appointment>
Assistant
type{"treatment", "operation"}: String
renewDiscount(adult)
Intern
internship_year{"second", "first"}: String
Overlapping inheritance
Patient{abstract}
NIP{unique}: String
/medical_card: MedicalCard
/child[0..1]:Child
/adult[0..1]:Adult
getDateOfBirth()
ConstraintValidator <AtLeastOneSubclassAndAge, Patient>
Child
parent_phone{max 12 characters}: String
Adult
discount{max = 50}: int
may not decrease
in persentage
Dynamic Inheritance
conduct>
1..*
MedicalCard
card_id{unique}:long
description{max 1000 characters}: String
/storage: Storage
/owner:Patient
in characters
<store
0..*
card_id{unique}:long
Qualified association
Storage
storage_id{unique}:long
type{max 10 characters}: String
capacity[0..1]: Double
/cards[0..1]: Map<Long, MedicalCard>
getCardById(long id)
in square meters
Composition
has>
book>
Association with attribute
Reservation
reservation_Id{unique}:long
reserve_date:Date
payment_method{"card", "cash"}: String
payment_date:Date
/adult: Adult
/appointment:Appointment
Appointment
app_id{unique}:long
description_disease{max 500 characters}: String
status {"registered","completed", "cancelled by customer"}:String
note[0..1]; String
appointment_date: Date
/patient: Adult
/treatments[0..*]: Set<Treatment>
/reservation:Reservation
getAdultAppointments(long id)
showAppointmentDetails(long id)
in characters
<assigned to
0..*
Treatment
treatment_Id{unique}:long
description{max 500 characters}: String
/appointments[0..*]:Set<Appointment>
deleteTreatment(long treatment id)

5. Use Case Scenario " View and Edit Appointments History " (as text):

In the scenario for "View and Edit Appointments History" use-case, we have a patient, who is an adult, wishing to view and edit his previous appointments using an online system. The system provides a process where the adult patient can select from a range of his previous appointments; after view the appointment info, appointment reservation info and treatments assigned to this appointment; delete treatment which patient does not need.

Before initiating the View and Edit Appointments History use case, it is required that at least one adult is already within the system. Once this condition is met, the adult patient can proceed with the following steps.

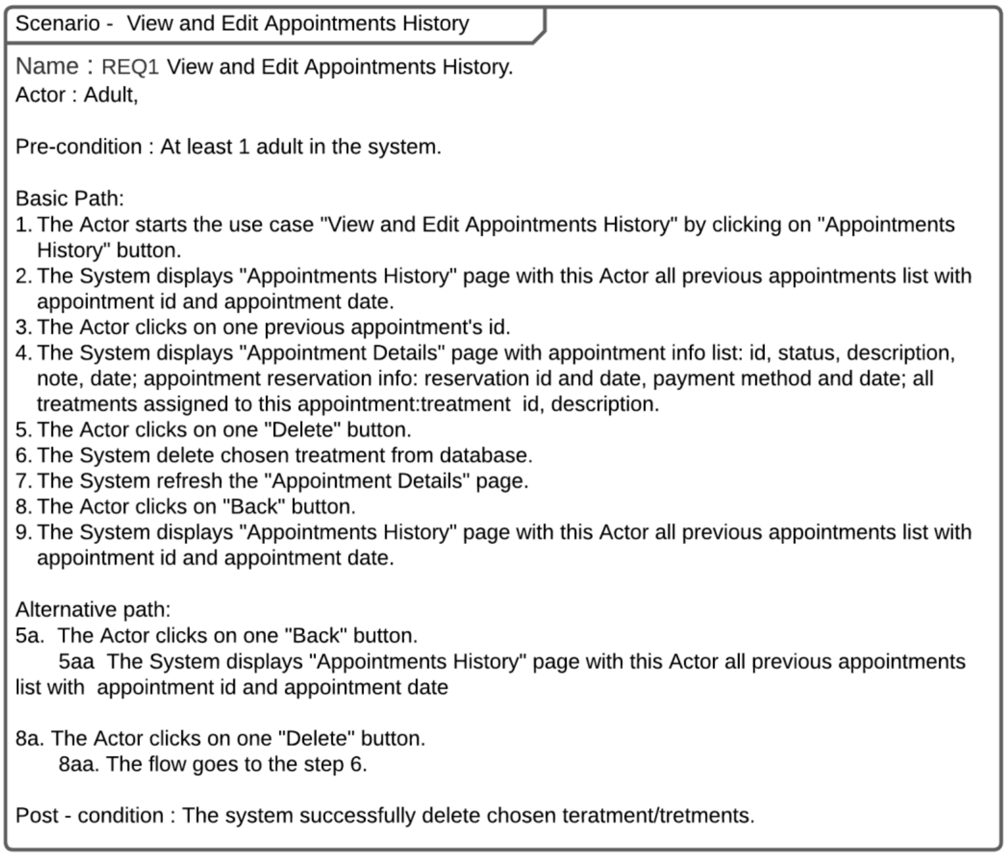
To begin, the patient initiates the actions by clicking on "Appointments History" button. This action triggers the system to present the patient the list of all his past appointments with appointment id and date, displayed on the "Appointments History" page. The patient then has the freedom to choose a specific past appointment id.

Once the patient has selected a desired past appointment, the system redirects him to the "Appointment Details" page. This page contains a list of appointment info like: id, status, description, note, date; appointment reservation info: reservation id and date, payment date and method; treatments assigned to this appointment: id and description. At "Appointment Details" page there are buttons like one "Back" and one or several "Delete".

At this step adult can decide go back to "Appointments History" page by clicking on "Back" button.

Or to click on "Delete" button which will leads to editing the information and to deletion of chosen treatment from database and after that the system will refresh the "Appointment Details"page view. The adult can delete/or not as many treatments as he has available at chosen appointment. After deleting the patient also can go back to "Appointments History" page by clicking on "Back" button. With this, the View and Edit Appointments History is successfully completed.

6. Use Case Scenario " View and Edit Appointments History":



Scenario - View and Edit Appointments History

Name : REQ1 View and Edit Appointments History.
Actor : Adult,

Pre-condition : At least 1 adult in the system.

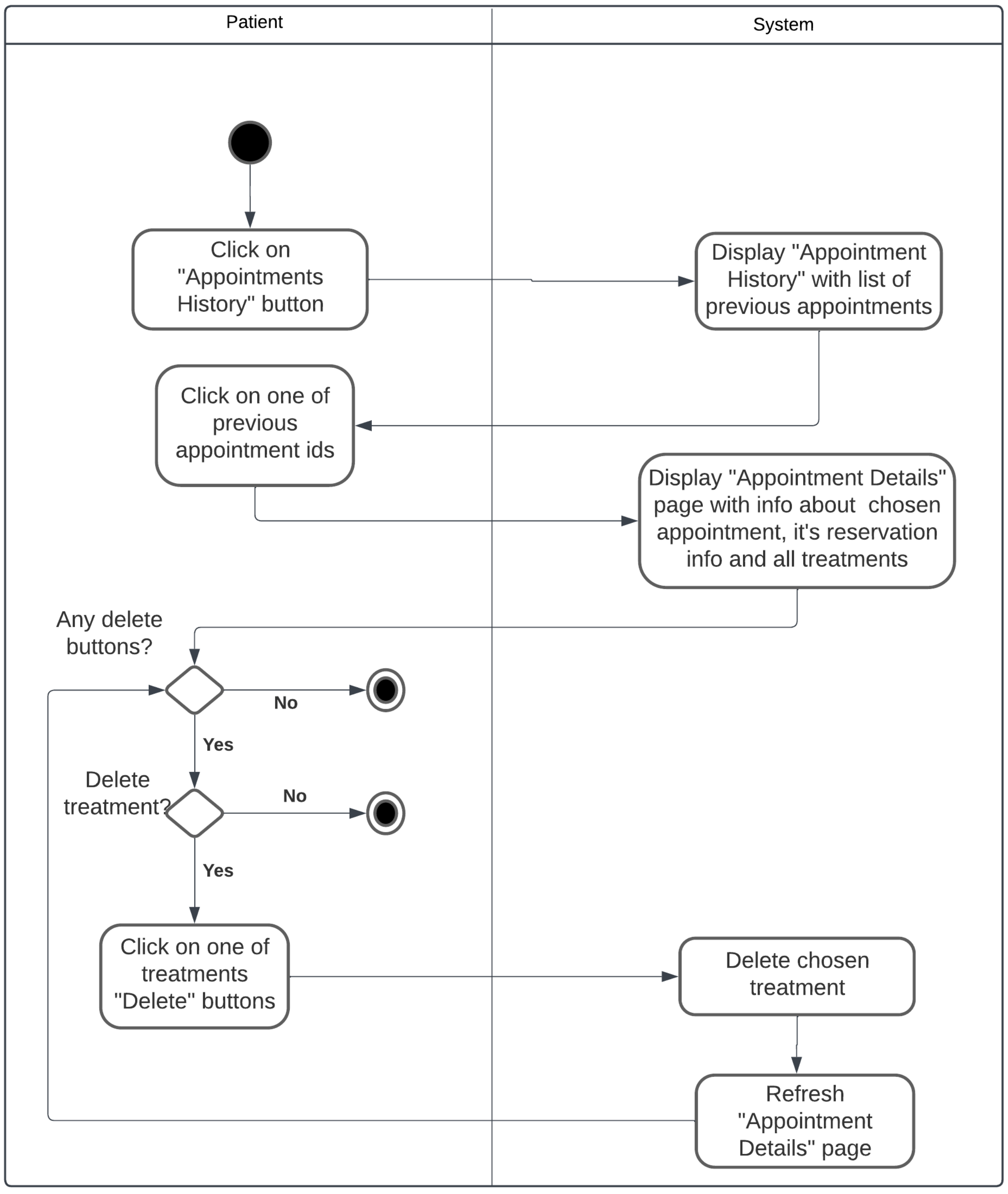
Basic Path:
1. The Actor starts the use case "View and Edit Appointments History" by clicking on "Appointments History" button.
2. The System displays "Appointments History" page with this Actor all previous appointments list with appointment id and appointment date.
3. The Actor clicks on one previous appointment's id.
4. The System displays "Appointment Details" page with appointment info list: id, status, description, note, date; appointment reservation info: reservation id and date, payment method and date; all treatments assigned to this appointment:treatment id, description.
5. The Actor clicks on one "Delete" button.
6. The System delete chosen treatment from database.
7. The System refresh the "Appointment Details" page.
8. The Actor clicks on "Back" button.
9. The System displays "Appointments History" page with this Actor all previous appointments list with appointment id and appointment date.

Alternative path:
5a. The Actor clicks on one "Back" button.
  5aa The System displays "Appointments History" page with this Actor all previous appointments list with appointment id and appointment date

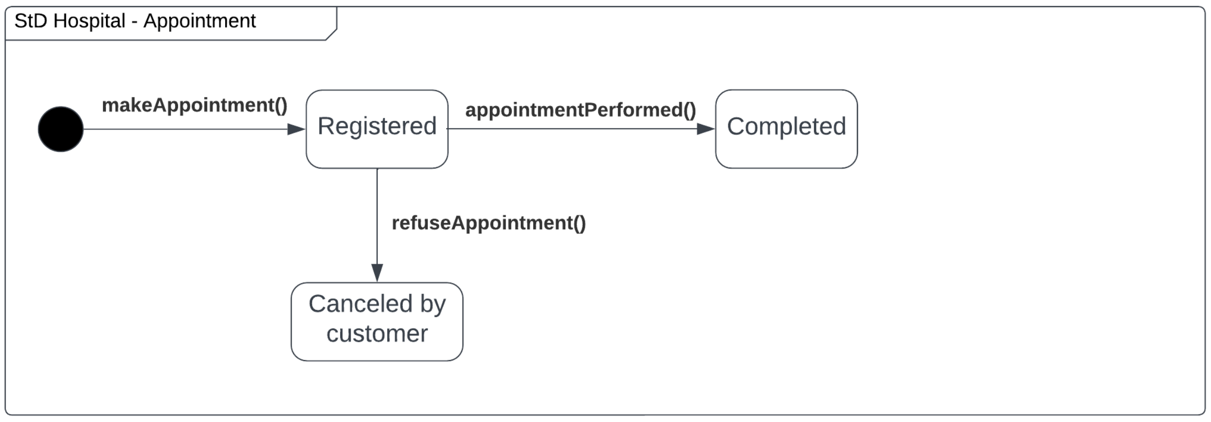
8a. The Actor clicks on one "Delete" button.
  8aa. The flow goes to the step 6.

Post - condition : The system successfully delete chosen teratment/tretments.

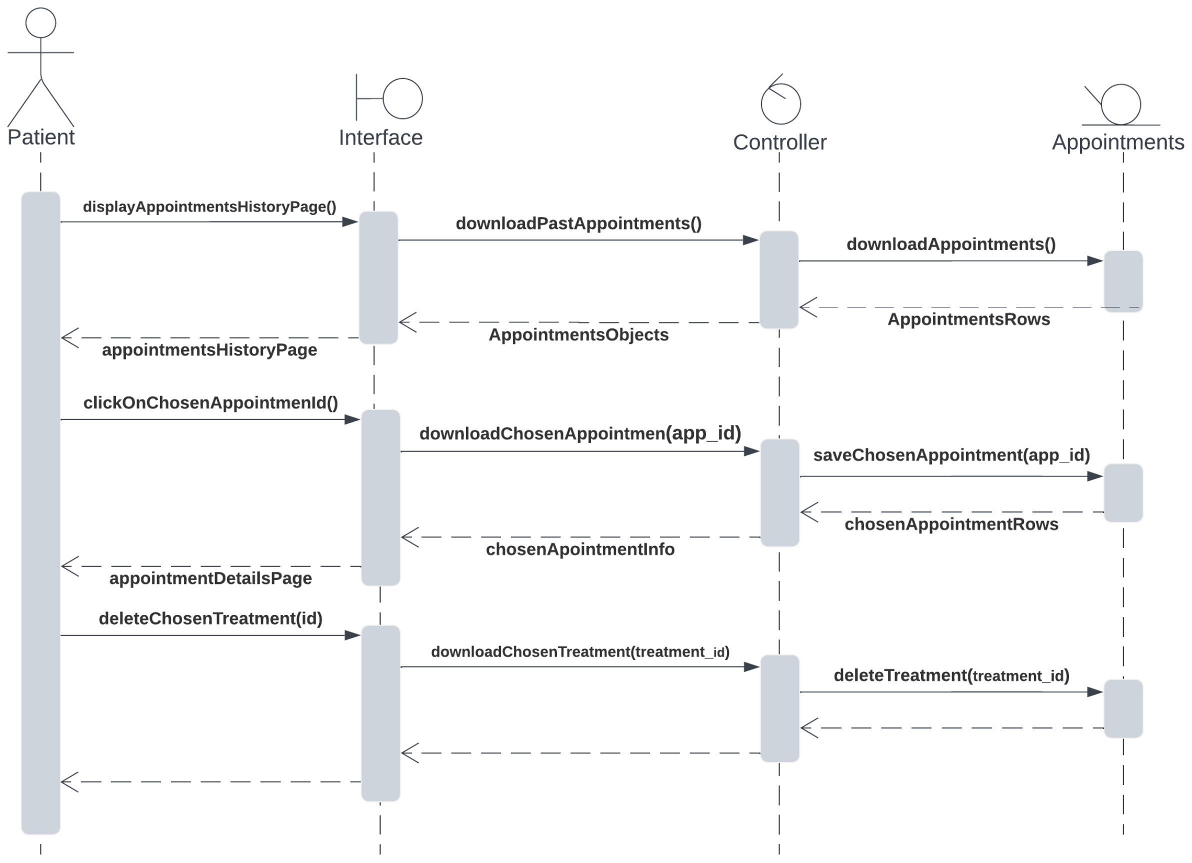
7. Activity diagram " View and Edit Appointments History":


Patient
System
Click on "Appointments History" button
Display "Appointment History" with list of previous appointments
Click on one of previous appointment ids
Display "Appointment Details" page with info about chosen appointment, it's reservation info and all treatments
Any delete buttons?
No
Yes
Delete treatment?
No
Yes
Click on one of treatments "Delete" buttons
Delete chosen treatment
Refresh "Appointment Details" page

8. State diagram class "Appointment":

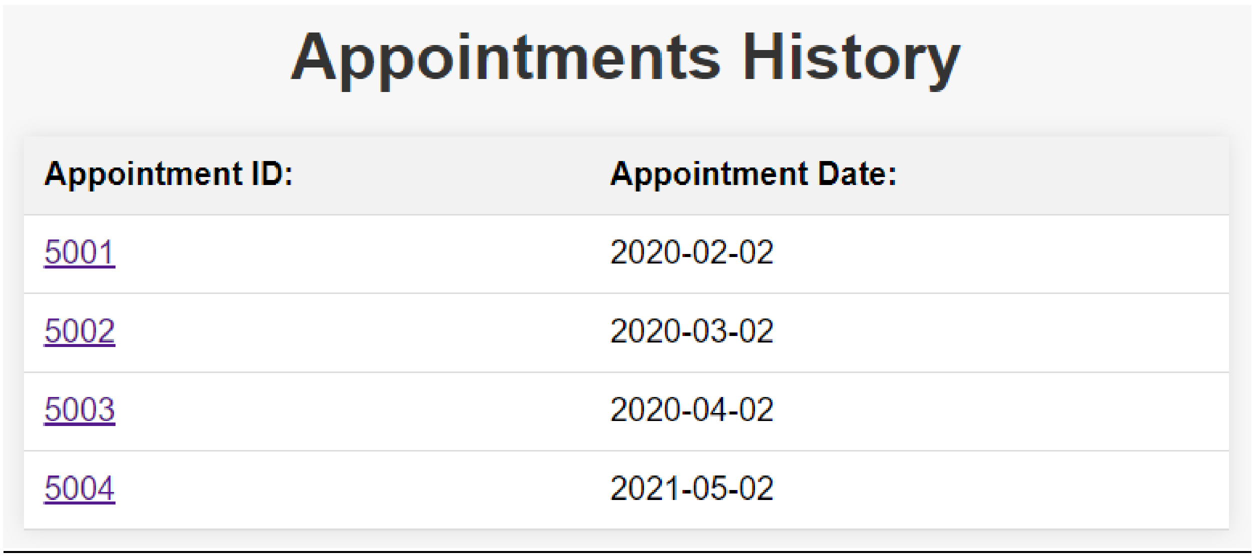


9. Sequence (interaction) diagram " View and Edit Appointments History":


Patient
Interface
Controller
Appointments
displayAppointmentsHistoryPage()
downloadPastAppointments()
downloadAppointments()
AppointmentsRows
AppointmentsObjects
appointmentsHistoryPage
clickOnChosenAppointmenId()
downloadChosenAppointmen(app_id)
saveChosenAppointment(app_id)
chosenAppointmentRows
chosenApointmentInfo
appointmentDetailsPage
deleteChosenTreatment(id)
downloadChosenTreatment(treatment_id)
deleteTreatment(treatment_id)

10. The GUI design of " View and Edit Appointments History":

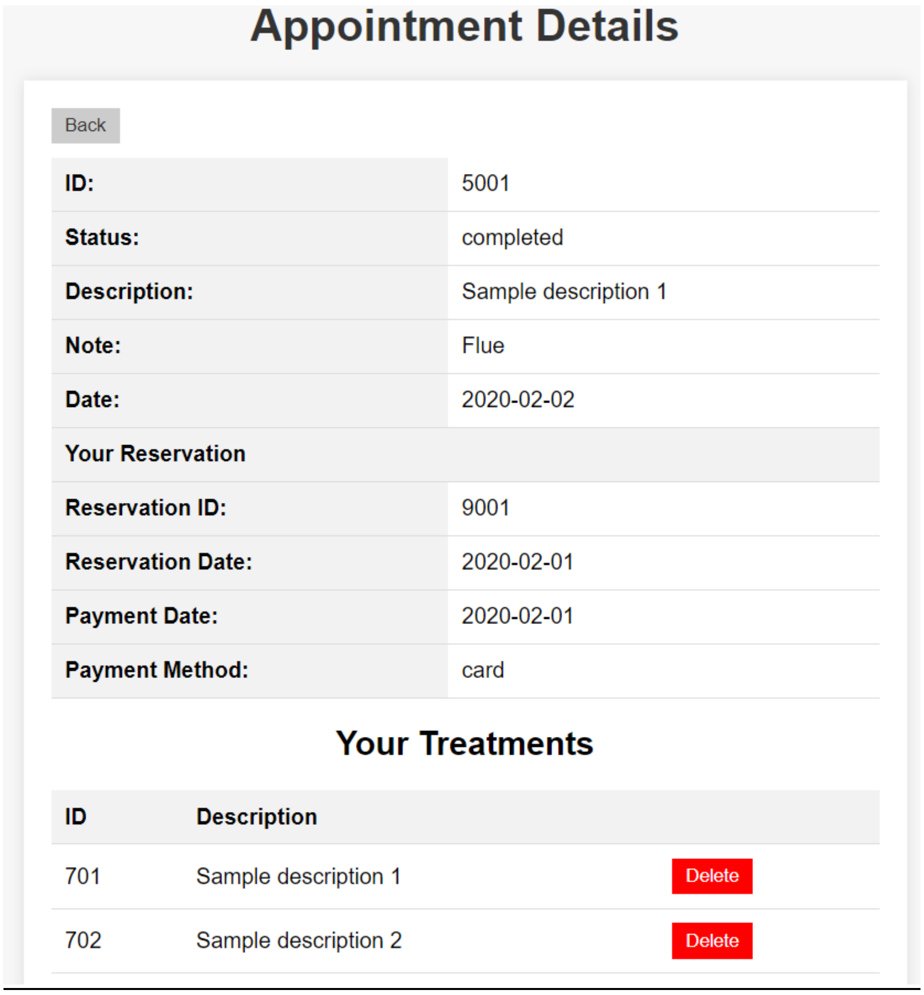
- *-Appointments History Page- Begin our use case here*



Appointments History

Appointment ID:	Appointment Date:
5001	2020-02-02
5002	2020-03-02
5003	2020-04-02
5004	2021-05-02

- *-Appointment Details Page- Page after clicking on appointment*



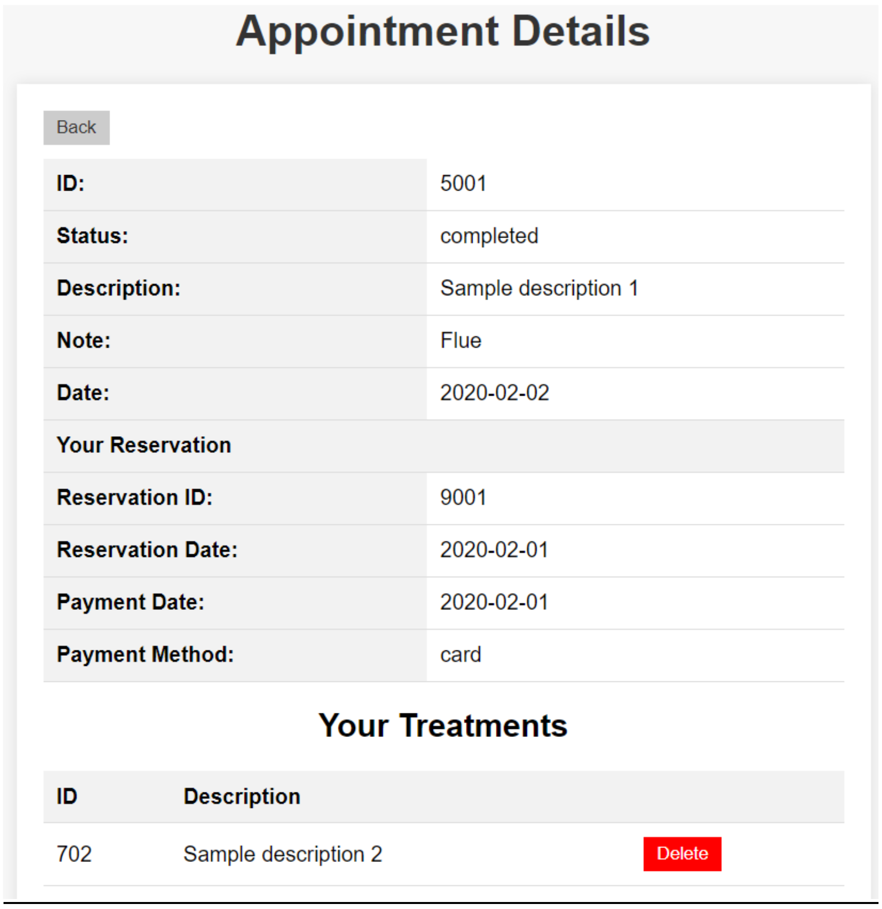
Appointment Details

ID:	5001
Status:	completed
Description:	Sample description 1
Note:	Flue
Date:	2020-02-02
Your Reservation	
Reservation ID:	9001
Reservation Date:	2020-02-01
Payment Date:	2020-02-01
Payment Method:	card

Your Treatments

ID	Description	
701	Sample description 1	Delete
702	Sample description 2	Delete

- *-Appointment Details Page- After deleting treatment*



Appointment Details

Back

ID:	5001
Status:	completed
Description:	Sample description 1
Note:	Flue
Date:	2020-02-02
Your Reservation	
Reservation ID:	9001
Reservation Date:	2020-02-01
Payment Date:	2020-02-01
Payment Method:	card

Your Treatments

ID	Description	
702	Sample description 2	Delete

11. The discussion of design decisions and the effect of dynamic analysis:

• The Class Diagram is designed for the implementation using an ORM framework-Hibernate in particular. All the classes are designed as models for ORM entities. This diagram describes the data types, association and some logic utilization classes with functions.

• Hibernate entities will be used in classes and they will need id which are presented in the diagram. Some private fields annotation, getters and setters can be added.

• Overlapping inheritance. It will be replaced by composition @OnoToOne between abstract class Employee and subclass Doctor, Assistant and Intern. Custom constraint for validation subclasses will be created to validate that at least one subclass is assigned and it can be more than one because of overlapping inheritance. Employee abstract class with annotation @AtLeastOneSubClass.

• Dynemic inheritance. It will be replaced by composition @OnoToOne between abstract class Patient and subclass Child and Adult. Custom constraint for validation subclasses will be created to validate that only one subclass is assigned but one subclass can become another one and have dynamic inheritance due to age(Child object after 18 years will assign its attributes to Adult object, and child old object will be deleted). Patient abstract class with annotation @AtLeastOneSubClassAnd Age.

- Classes Adult and Appointment has association with attribute Reservation, So I will use @OneToMany relationship between Adult and Reservation. And @OneToOne association between Appointment and Reservation.

- Classes Appointment and Treatment has @ManyToMany relationship that will cause to create additional table appointment_treatment(app_id,treatment_id) in database in order to handle many to many relationship.